***Analysis Report***

***Deep Learning Challenge***

***Module 21***

**Overview**

For the **Deep Learning Challenge**, Module 21, I constructed a neural networks tool for the non-profit foundation Alphabet Soup. Specifically, I use the features of a dataset provided, **charity\_data.csv**, to create a binary classifier that can predict whether applicants will be successful in their ventures if granted the requested funding. Note, ‘neural networks’ is an advanced form of machine learning called ‘deep learning’, that uses interconnected nodes or neurons in a layered structure that resembles the human brain – *which is, dare I say, pretty awesome*!

**Results**

Here follows a summary of results, organized in order of the neural networks process. Note, this process summary directly follows that of my 4th optimization attempt (***AphabetSoupCharity\_Optimization\_4.ipynb***), for which I was able to meet the ‘greater than 75%’ accuracy goal.

**Data Preprocessing**

* The target variable was identified to be **IS\_SUCCESSFUL.**
* Variables identified as features included the following:

**NAME**

**APPLICATION\_TYPE**

**AFFILIATION**

**CLASSIFICATION**

**USE\_CASE**

**ORGANIZATION**

**STATUS**

**INCOME\_AMT**

**SPECIAL\_CONSIDERATIONS**

**ASK\_AMT**

* In my original construction, **NAME** was not included here, but dropped. Yet, it was not until I added it back and binned it that I reached the accuracy goal.
* **EIN** was neither a target nor feature, and thus removed.

**Compiling, Training, and Evaluating the Model**

I offer the following chart with a breakdown of the model attributes I employed, from initial model construction (***AlphabetSoupCharity.ipynb***), to the final optimization that reached accuracy goal (***AphabetSoupCharity\_Optimization\_4.ipynb***) – highlighted in green 😊. As documented below, I experimented with the adjusting of variables removed, variables binned, the number of neurons and layers, activation function and even the number of epochs, comparing the ‘predictive accuracy’ of each trial.



* I achieved target model performance in my last optimization attempt – 0.7925 😊.
* Steps taken in attempts to achieve model performance were as follow:
* Preprocessing – Variables were identified for removal, and those kept were identified for binning. I began by removing both **EIN** and **NAME** and binning only **APPICATION\_TYPE** and **CLASSIFICATION**.
* Compile, Train, and Evaluate – I experimented by adding additional layers, increasing neuron counts, shifting to a more complex activation function (from ‘relu’ to ‘sigmoid’), and applying this activation function to more layers. I even adjusted the number of epochs – all with little change in accuracy.
* Success! Success finally came when I made a rather broad change in ‘preprocessing’, adding back **NAME**, and then binning it. Keeping all other attributes the same, including those in the Compile & Train stages, the model showed an abrupt improvement – far surpassing the 72% it was stuck at, and even passing the 75% goal, ultimately reaching a merit-worthy 79% 😊.

**Summary Conclusion**

In conclusion, the Alphabet Soup foundation can benefit from utilizing the model built with my fourth optimization attempt (output file: ***AlphabetSoupCharity\_Optimization\_4.h5***). Applying this model to new large datasets will determine which applicants will be successful, and therefore deserving of funds, almost 8 times out 10 (accuracy = 0.7925) – *far better, and faster, than a simple coin toss* 😊.

My recommendation for further investigation and testing would be trying the **random forest** model. It turns out that in some cases the **random forest** model outperforms a **neural networks** model. In particular, when the data is ‘relatively’ small (*relative in terms of BIG data*) and not complicated, as with our dataset, the **random forest** will likely be surpass the **neural networks** model in its predictive power.